Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Specification Update

Release Date: October 2002

Order Number: 251309-004

The Mobile Intel® Celeron® Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

CONTENTS

REVISION HISTORY....ii
PREFACE....iii
GENERAL INFORMATION....1
IDENTIFICATION INFORMATION....2
SUMMARY OF CHANGES....4
ERRATA....8
DOCUMENTATION CHANGES....21
SPECIFICATION CLARIFICATIONS....42
SPECIFICATION CHANGES....43

REVISION HISTORY

Date of Revision	Version	Description
June 2002	-001	Initial Release
August 2002	-002	Updated Identification Information table.
September 2002	-003	Updated Documentation Change summary by removing old items that have been incorporated into the Software Developer's Manual; Added errata V30-V32; Added Documentation Changes V3-V24.
October 2002	-004	Updated Summary of Changes; Added prefix letter W; Removed previous errata V32 as not applicable, added new errata V32; Added Documentation Changes V25-V32.

PREFACE

This document is an update to the specifications contained in the following document:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Numbers 251308)
- *Intel® Architecture Software Developer's Manual, Volumes 1, 2, and 3* (Order Numbers 243190, 243191, and 243192, respectively)

It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains S-Specs, Errata, Documentation Changes, Specification Clarifications and Specification Changes.

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel Pentium 4 processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that processor are present on all devices unless otherwise noted.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

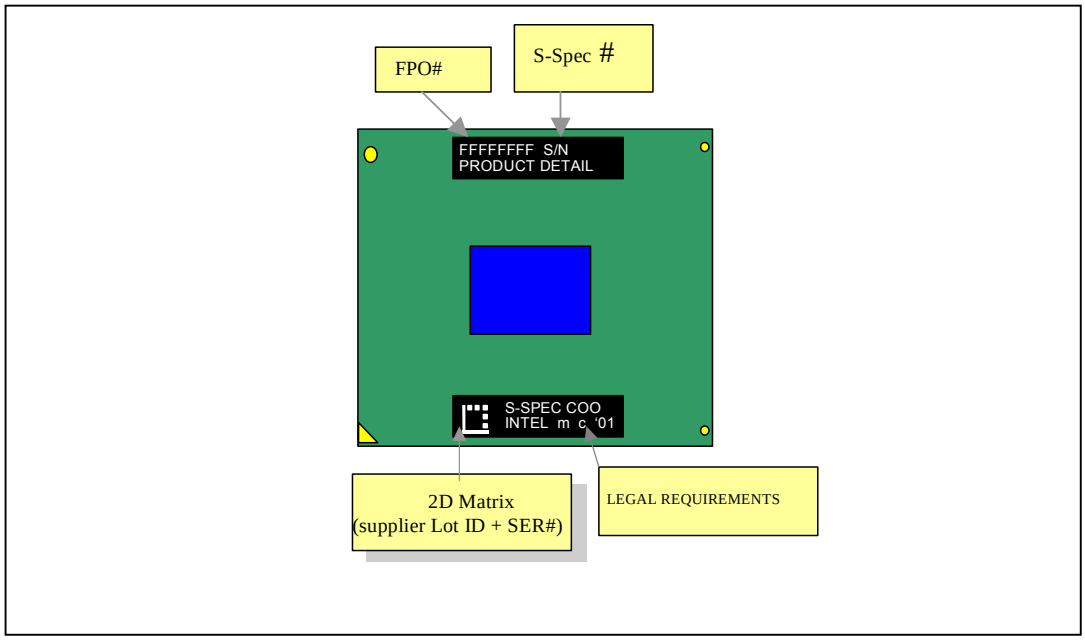
Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Specification Changes are modifications to the current published specifications for the Intel Celeron processor. These changes will be incorporated in the next release of the specifications.

Mobile Intel® Celeron® Processor on .13 Micron Process and in Micro-FCPGA Package Specification Update

GENERAL INFORMATION

Mobile Intel® Celeron® Processor on .13 Micron Process (Micro-FCPGA) Markings



IDENTIFICATION INFORMATION

The Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA package can be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
1111	0010	00001000[4]
1111	0010	00001111[5]

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after Reset, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after Reset, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.
4. This brand id MUST be used in conjunction with CPUID = 0F24h.
5 This brand id MUST be used in conjunction with CPUID = 0F27h.

Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Identification Information

S-Spec	Product Stepping	L2 Cache Size (bytes)	CPU Signature	Core Frequency	Bus Frequency	Voltage	Package	Notes
SL6FM	B0	256 K	0F24h	1.4 GHz	400 MHz	1.3 V	Micro-FCPGA	1
SL6FN	B0	256 K	0F24h	1.5 GHz	400 MHz	1.3 V	Micro-FCPGA	1

NOTES:

1. Based on Bo-Shrink process.

SUMMARY OF CHANGES

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Intel Pentium 4 processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Erratum, Documentation Change, Specification Clarification, or Specification Change applies to the given processor stepping.
(No mark) or (blank box):	This item is fixed in or does not apply to the given stepping.
PlanFix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Doc:	Document change or update that will be implemented.
PKG:	This column refers to errata on the Intel® Pentium® 4 processor substrate.
AP:	APIC related erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor

B = Mobile Intel® Pentium® II processor

C = Intel® Celeron® processor

D = Intel® Pentium® II Xeon™ processor

E = Intel® Pentium® III processor

G = Intel® Pentium® III Xeon™ processor

H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz

K = Mobile Intel® Pentium® III processor-M

M = Mobile Intel® Celeron® processor

N = Intel® Pentium® 4 processor

P = Intel® Xeon™ processor

T = Mobile Intel® Pentium® 4 processor-M

V = Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package

W = Low Voltage Intel® Xeon™ processor

The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

SUMMARY OF ERRATA

NO.	B0	Plans	ERRATA
V1	X	NoFix	I/O restart in SMM may fail after simultaneous machine check exception (MCE)
V2	X	NoFix	MCA registers may contain invalid information if RESET# occurs and PWRGOOD is not held asserted
V3	X	NoFix	Transaction is not retried after BINIT#
V4	X	NoFix	Invalid opcode 0FFFh requires a ModRM byte
V5	X	NoFix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
V6	X	NoFix	The processor flags #PF instead of #AC on an unlocked CMPXCHG8B instruction
V7	X	NoFix	When in no-fill mode the memory type of large pages are incorrectly forced to uncacheable
V8	X	NoFix	Processor may hang due to speculative page walks to non-existent system memory
V9	X	NoFix	The IA32_MC1_STATUS register may contain incorrect information for correctable errors
V10	X	NoFix	Debug mechanisms may not function as expected
V11	X	NoFix	Machine check architecture error reporting and recovery may not work as expected
V12	X	NoFix	Cascading of performance counters does not work correctly when forced overflow is enabled
V13	X	NoFix	EMON event counting of x87 loads may not work as expected
V14	X	NoFix	Speculative page fault may cause livelock
V15	X	NoFix	Incorrect data may be returned when page tables are in Write Combining (WC) memory space
V16	X	NoFix	Processor issues inconsistent transaction size attributes for locked operation
V17	X	PlanFix	Multiple accesses to the same S-state L2 cache line and ECC error combination may result in loss of cache coherency
V18		Fixed	Processor may hang when resuming from Deep Sleep state
V19	X	NoFix	When the processor is in the System Management Mode (SMM), debug registers may be fully writeable
V20	X	NoFix	Associated counting logic must be configured when using Event Selection Control (ESCR) MSR
V21	X	NoFix	IA32_MC0_ADDR and IA32_MC0_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error
V22	X	PlanFix	CR2 May Be Incorrect or an Incorrect Page Fault Error Code May Be Pushed Onto Stack After Execution of an LSS Instruction
V23	X	PlanFix	BPM[5:3]# and GHI# V_{IL} Do Not Meet Specification
V24	X	NoFix	Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle
V25	X	NoFix	System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction

SUMMARY OF ERRATA

NO.	B0	Plans	ERRATA
			to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
V26	X	PlanFix	L2 Cache May Contain Stale Data in the Exclusive State
V27	X	PlanFix	Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may Cause IO and Special Cycle Failure
V28	X	PlanFix	Erroneous BIST Result Found in EAX Register After Reset
V29	X	PlanFix	Processor Does not Flag #GP on Non-zero Write to Certain MSRs
V30	X	NoFix	Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch
V31	X	PlanFix	Processor Does not Respond to Break Requests From ITP
V32	X	Fixed	Processor Signature Returns Incorrect Number of ITLB Entries

SUMMARY OF DOCUMENTATION CHANGES

NO.	B0	Plans	DOCUMENTATION CHANGES
V1	X	Doc	SSE and SSE2 Instructions Opcodes
V2	X	Doc	Executing the SSE2 Variant on a Non-SSE2 Capable Processor
V3	X	Doc	Direction Flag (DF) Mistakenly Denoted as a System Flag
V4	X	Doc	Fopcode Compatibility Mode
V5	X	Doc	FCOS, FPTAN, FSIN, and FSINCOS Trigonometric Domain not Correct
V6	X	Doc	Incorrect Description of Stack
V7	X	Doc	EFLAGS Register Correction
V8	X	Doc	PSE-36 Paging Mechanism
V9	X	Doc	0x33 Opcode
V10	X	Doc	Incorrect Information for SLDT
V11	X	Doc	LGDT/LIDT Instruction Information Correction
V12	X	Doc	Errors in Instruction Set Reference
V13	X	Doc	RSM Instruction Set Summary
V14	X	Doc	Correct MOVAPS and MOVAPD Operand Section
V15	X	Doc	DAA—Decimal Adjust AL after Addition

SUMMARY OF DOCUMENTATION CHANGES

NO.	B0	Plans	DOCUMENTATION CHANGES
V16	X	Doc	DAS—Decimal Adjust AL after Subtraction
V17	X	Doc	Omission of Dependency Between BTM and LBR
V18	X	Doc	I/O Permissions Bitmap Base Addy > 0xDFFF Does not Cause #GP(0) Fault
V19	X	Doc	Wrong Field Width for MINSS and MAXSS
V20	X	Doc	Figure 15-12 PEBS Record Format
V21	X	Doc	I/O Permission Bit Map
V22	X	Doc	Cache Description
V23	X	Doc	Instruction Formats and Encoding
V24	X	Doc	Machine-Check Initialization
V25	X	Doc	Incorrect Description of MOVAPS #UD
V26	X	Doc	PSE-36 Paging Mechanism
V27	X	Doc	Segment Wraparound Compatibility
V28	X	Doc	Performance Counting Clocks
V29	X	Doc	New Cascading Counters Chapter
V30	X	Doc	Updated Table B-1 in Appendix B
V31	X	Doc	Message Signaled Interrupts
V32	X	Doc	Selecting Memory Types for Pentium 4, Intel Xeon, and Pentium III Processors

SUMMARY OF SPECIFICATION CLARIFICATIONS

NO.	B0	Plans	SPECIFICATION CLARIFICATIONS
			There are no Specification Clarifications

SUMMARY OF SPECIFICATION CHANGES

NO.	B0	Plans	SPECIFICATION CHANGES
			There are no Specification Changes

ERRATA

V1. I/O Restart in SMM may Fail after Simultaneous Machine Check Exception (MCE)

Problem: If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.

Implication: A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler may attempt to restart such an I/O instruction, but will have an incorrect state due to the MCE handler call, leading to failure of the restart and shutdown of the processor.

Workaround: If a system implementation must support both SMM and board I/O restart, the first thing the SMM handler code should do is check for a pending MCE. If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the MCE handler to execute. If there is no MCE pending, the SMM handler may proceed with its normal operation.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V2. MCA Registers may Contain Invalid Information if RESET# Occurs and PWRGOOD is not Held Asserted

Problem: This erratum can occur as a result either of the following events:

- PWRGOOD is de-asserted during a RESET# assertion causing internal glitches that may result in the possibility that the MCA registers latch invalid information.
- Or during a reset sequence if the processor's power remains valid regardless of the state of PWRGOOD, and RESET# is re-asserted before the processor has cleared the MCA registers, the processor will begin the reset process again but may not clear these registers.

Implication: When this erratum occurs, the information in the MCA registers may not be reliable.

Workaround: Ensure that PWRGOOD remains asserted throughout any RESET# assertion and that RESET# is not re-asserted while PWRGOOD is de-asserted.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V3. Transaction is not Retried after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, it will not be retried.

Implication: When this erratum occurs, locked transactions will unexpectedly not be retried.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V4. Invalid Opcode 0FFFh Requires a ModRM Byte

Problem: Some invalid opcodes require a ModRM byte (or other following bytes), while others do not. The invalid opcode 0FFFh did not require a ModRM byte in previous generation Intel architecture processors, but does in the Intel® Pentium® 4 processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Intel® Pentium® 4 processor.

Workaround: Use a ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V5. FSW may not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64Kbyte or 4Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64Kbyte or 4Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V6. The Processor Flags #PF Instead of #AC on an Unlocked CMPXCHG8B Instruction

Problem: If a data page fault (#PF) and alignment check fault (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.

Implication: Software that depends #AC before #PF will be affected since #PF is flagged in this case.

Workaround: Remove the software's dependency on the fact that #AC has precedence over #PF. Alternately, reload the page in the page fault handler and then restart the faulting instruction.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V7. When in No-Fill Mode the Memory Type of Large Pages are Incorrectly Forced to Uncacheable

Problem: When the processor is operating in No-Fill Mode (CR0.CD=1), the paging hardware incorrectly forces the memory type of large (PSE-4M and PAE-2M) pages to uncacheable (UC) memory type regardless of the MTRR settings. By forcing the memory type of these pages to UC, load operations, which should hit valid data in the L1 cache, are forced to load the data from system memory. Some applications will lose the performance advantage associated with the caching permitted by other memory types.

Implication: This erratum may result in some performance degradation when using no-fill mode with large pages.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V8. Processor may Hang due to Speculative Page Walks to Non-Existent System Memory

Problem: A load operation that misses the Data Translation Lookaside Buffer (DTLB) will result in a page-walk. If the page-walk loads the Page Directory Entry (PDE) from cacheable memory and that PDE load returns data that points to a valid Page Table Entry (PTE) in uncacheable memory the processor will access the address referenced by the PTE. If the address referenced does not exist the processor will hang with no response from system memory.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space that are marked valid must point to physical addresses that will return a data response to the processor.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V9. The IA32_MC1_STATUS Register may Contain Incorrect Information for Correctable Errors

Problem: When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_STATUS register may be updated with incorrect information. The IA32_MC1_STATUS register should not be updated for speculative loads.

Implication: When this erratum occurs, the IA32_MC1_STATUS register will contain incorrect information for correctable errors.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V10. Debug Mechanisms may not Function as Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers, e.g., LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V11. Machine Check Architecture Error Reporting and Recovery may not Work as Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, IA32_MC0_STATUS.UNCOR and IA32_MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because IA32_MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all IA32_MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR register. In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- The local xAPIC has an Error Status Register which records all errors which it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it received. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.
- If an instruction fetch results in an uncorrectable error and there is also a debug breakpoint at this address, the processor will livelock and the uncorrectable error will not be logged in the machine check registers.
- The MCA Overflow bit should be set when an uncorrectable error resides within the register bank (valid bit is already set) and any subsequent errors occur. The Overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA Overflow bit will not be updated; thereby incorrectly indicating only one error has been received.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V12. Cascading of Performance Counters does not work Correctly when Forced Overflow is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V13. EMON Event Counting of x87 Loads may not Work as Expected

Problem: If a performance counter is set to count x87 loads and floating point exceptions are unmasked, the FPU Operand Data Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, the FPU Operand Data Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V14. Speculative Page Fault may cause Livelock

Problem: If the processor detects a page fault which is corrected before the operating system page fault handler can be called e.g. DMA activity modifies the page tables and the corrected page tables are left in a non-accessed or not dirty state, the processor may livelock. Intel has not been able to reproduce this erratum with commercial software.

Implication: Should this erratum be encountered the processor will livelock resulting in a system hang or operating system failure.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V15. Incorrect Data May be Returned When Page Tables Are In Write Combining Memory (WC) Space

Problem: If page directories and/or page tables are located in Write Combining (WC) memory, speculative loads to cacheable memory may complete with incorrect data.

Implication: Cacheable loads to memory mapped using page tables located in write combining memory may return incorrect data. Intel has not been able to reproduce this erratum with commercially available software.

Workarounds: Do not place page directories and/or page tables in WC memory.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V16. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V17. Multiple Accesses to the Same S-State L2 Cache Line and ECC Error Combination May Result in Loss of Cache Coherency

Problem: When a Read for Ownership (RFO) cycle has a 64 bit address match with an outstanding read hit on a line in the L2 cache which is in the S-state AND that line contains an ECC error, the processor should recycle the RFO until the ECC error is handled. Due to this erratum, the processor does not recycle the RFO and attempts to service both the RFO and the read hit at the same time.

Implication: When this erratum occurs, cache may become incoherent.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V18. Processor May Hang When Resuming From Deep Sleep State

Problem: When resuming from the Deep Sleep state the address strobe signals (ADSTB[1:0]#) may become out of phase with respect to the system bus clock (BCLK).

Implication: When this erratum occurs, the processor will hang.

Workaround: The system BIOS should prevent the processor from going to the Deep Sleep state.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V19. When the Processor is in the System Management Mode (SMM), Debug Registers May be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V20. Associated Counting Logic Must be Configured When Using Event Selection Control (ESCR) MSR

Problem: ESCR MSRs allow software to select specific events to be counted, with each ESCR usually associated with a pair of performance counters. ESCRs may also be used to qualify the detection of at-retirement events that support precise-event-based sampling (PEBS). A number of performance metrics that support PEBS require a 2nd ESCR to tag uops for the qualification of at-retirement events. (The first ESCR is required to program the at-retirement event.) Counting is enabled via counter configuration control registers (CCCR) while the event count is read from one of the associated counters. When counting logic is configured for the subset of at-retirement events that require a 2nd ESCR to tag uops, at least one of the CCCRs in the same group of the 2nd ESCR must be enabled.

Implication: If no CCCR/counter is enabled in a given group, the ESCR in that group that is programmed for tagging uops will have no effect. Hence a subset of performance metrics that require a 2nd ESCR for tagging uops may result in 0 count.

Workaround: Ensure that at least one CCCR/counter in the same group as the tagging ESCR is enabled for those performance metrics that require two ESCRs and tagging uops for at-retirement counting.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V21. IA32_MC0_ADDR and IA32_MC0_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error

Problem: If the processor experiences a data, address, or response parity error, the ADDRV and MISCV bits of the IA32_MC0_STATUS register are set, but the IA32_MC0_ADDR and IA32_MC0_MISC registers are not loaded with data regarding the error.

Implication: When this erratum occurs, the IA32_MC0_ADDR and IA32_MC0_MISC registers will contain invalid or stale data.

Workaround: Ignore any information in the IA32_MC0_ADDR and IA32_MC0_MISC registers after a data, address or response parity error.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

V22. CR2 May Be Incorrect or an Incorrect Page Fault Error Code May Be Pushed onto Stack After Execution of an LSS Instruction

Problem: Under certain timing conditions, the internal load of the selector portion of the LSS instruction may complete with potentially incorrect speculative data before the load of the offset portion of the address completes. The incorrect data is corrected before the completion of the LSS instruction but the value of CR2 and the error code pushed on the stack are reflective of the speculative state. Intel has not observed this erratum with commercially available software.

Implication: When this erratum occurs, the contents of CR2 may be off by two, or an incorrect page fault error code may be pushed onto the stack.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V23. BPM[5:3]# and GHI# VIL Does Not Meet Specification

Problem: The V_{IL} for BPM[5:3]# and GHI# is specified as 0.9 * GTLREF [V]. Due to this erratum the V_{IL} for these signals is 0.9 * GTLREF - .075 [V].

Implication: The processor requires a lower input voltage than specified to recognize a low voltage on the BPM[5:3]# and GHI# signals.

Workaround: When intending to drive the BPM[5:3]# or GHI# signals low, ensure that the system provides a voltage lower than 0.9 * GTLREF - .075 [V].

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V24. Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V25. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V26. L2 Cache May Contain Stale Data in the Exclusive State

Problem: If a cacheline (A) is in Modified (M) state in the write-combining (WC) buffers and in the Invalid (I) state in the L2 cache and its adjacent sector (B) is in the Invalid (I) state and the following scenario occurs:

1. A read to B misses in the L2 cache and allocates cacheline B and its associated second-sector pre-fetch into an almost full bus queue,
2. A Bus Read Line (BRL) to cacheline B completes with HIT# and fills data in Shared (S) state,
3. The bus queue full condition causes the prefetch to cacheline A to be cancelled, cacheline A will remain M in the WC buffers and I in the L2 while cacheline B will be in the S state.

Then, if the further conditions occur:

1. Cacheline A is evicted from the WC Buffers to the bus queue which is still almost full,

2. A hardware prefetch Read for Ownership (RFO) to cacheline B, hits the S state in the L2 and observes cacheline A in the I state, allocates both cachelines,
3. An RFO to cacheline A completes before the WC Buffers write modified data back, filling the L2 with stale data,
4. The writeback from the WC Buffers completes leaving stale data, for cacheline A, in the Exclusive (E) state in the L2 cache.

Implication: Stale data may be consumed leading to unpredictable program execution. Intel has not been able to reproduce this erratum with commercial software.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V27. Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may Cause IO and Special Cycle Failure

Problem: Remapping the APIC base address from its default can cause conflicts with either I/O or special cycle bus transactions.

Implication: Either I/O or special cycle bus transactions can be redirected to the APIC, instead of appearing on the front-side bus.

Workaround: Use any APIC base addresses above 0xDC001000 as the relocation address.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V28. Erroneous BIST Result Found in EAX Register After Reset

Problem: The processor may show an erroneous BIST (built-in self test) result in the EAX register bit 0 after reset.

Implication: When this erratum occurs, an erroneous BIST failure will be reported in the EAX register bit 0, however this failure can be ignored since it is not accurate.

Workaround: It is possible for BIOS to workaround this issue by masking off bit 0 in the EAX register where BIST results are written.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V29. Processor Does not Flag #GP on Non-zero Write to Certain MSRs

Problem: When a non-zero write occurs to the upper 32 bits of IA32_CR_SYSENTER_EIP or IA32_CR_SYSENTER_ESP, the processor should indicate a general protection fault by flagging #GP. Due to this erratum, the processor does not flag #GP.

Implication: The processor unexpectedly does not flag #GP on a non-zero write to the upper 32 bits of IA32_CR_SYSENTER_EIP or IA32_CR_SYSENTER_ESP. No known commercially available operating system has been identified to be affected by this erratum.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V30. Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch

Problem: If A20M# and INIT# are simultaneously asserted by software, followed by a data access to the 0xFFFFFXXX memory region, with A20M# still asserted, incorrect data will be accessed. With A20M# asserted, an access to 0xFFFFFXXX should result in a load from physical address 0xFFEFFXXX. However, in the case of A20M# and INIT# being asserted together, the data load will actually be from the physical address 0xFFFFFXXX. Code accesses are not effected by this erratum.

Implication: Processor may fetch incorrect data, resulting in BIOS failure.

Workaround: Deasserting and reasserting A20M# prior to the data access will workaround this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V31. Processor Does not Respond to Break Requests From ITP

Problem: On power-up and low-power state transitions, the processor's TAP circuitry may remain in the Tap-Logic-Reset(TLR) state.

Implication: The ITP is unable to cause a break on reset in the processor, which may prevent the loading of processor and chipset registers, or affect the ability to debug from cold boot and low power transitions.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

V32. CPUID Instruction Returns Incorrect Number of ITLB Entries

Problem: When CPUID is executed with EAX = 2 it should return a value of 51h in EAX[15:8] to indicate that the Instruction Translation Lookaside Buffer (ITLB) has 128 entries. Due to this erratum, the processor returns 50h (64 entries).

Implication: Software may incorrectly report the number of ITLB entries. Operation of the processor is not affected.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

DOCUMENTATION CHANGES

The Documentation Changes listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Numbers 251308)
- *Intel® Architecture Software Developer's Manual, Volumes 1, 2, and 3* (Order Numbers 243190, 243191, and 243192, respectively)

All Documentation Changes will be incorporated into a future version of the appropriate Mobile Intel Celeron Processor on 0.13 Micron Process in Micro-FCPGA Package documentation.

V1. SSE and SSE2 Instructions Opcodes

The note at the end of section 2.2 in the *Intel Architecture Software Developer's Manual*, *Vol 2*: *Instruction Set Reference* states:

> NOTE:
> Some of the SSE and SSE2 instructions have three-byte opcodes. For these three-byte opcodes, the third opcode byte may be F2H, F3H, or 66H. For example, the SSE2 instruction CVTDQ2PD has the three-byte opcode F3 OF E6. The third opcode byte of these three-byte opcodes should not be thought of as a prefix, even though it has the same encoding as the operand size prefix (66H) or one of the repeat prefixes (F2H and F3H). As described above, using the operand size and repeat prefixes with SSE and SSE2 instructions is reserved.

It should state:

> NOTE:
> Some of the SSE and SSE2 instructions have three-byte opcodes. For these three-byte opcodes, the third opcode byte may be F2H, F3H, or 66H. For example, the SSE2 instruction CVTDQ2PD has the three-byte opcode F3 OF E6. The third opcode byte of these three-byte opcodes should not be thought of as a prefix, even though it has the same encoding as the operand size prefix (66H) or one of the repeat prefixes (F2H and F3H). As described above, using the operand size and repeat prefixes with SSE and SSE2 instructions is reserved. It should also be noted that execution of SSE2 instructions on an Intel processor that does not support SSE2 (CPUID Feature flag register EDX bit 26 is clear) will result in unpredictable code execution.

V2. Executing the SSE2 Variant on a Non-SSE2 Capable Processor

In Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference the section for each of the following instructions states that executing the instruction in real or protected mode on a processor for which the SSE2 feature flag returned by CPUID is 0 (SSE2 not supported by the processor) will result in the generation of an undefined opcode exception (#UD). This is incorrect. The SSE2 form of these instructions is defined by opcodes for which the leading opcode byte maps into an operand size prefix. Executing the SSE2 variant of these instructions on a non-SSE2 capable processor will result in an SSE like operation and not a #UD. Refer to section 2.2 of the Intel Architecture Software Developer's Manual, Vol 2 for more detail.

Instructions:

MOVD xmm, r32; MOVD r32, xmm; MOVDQA; MOVDQU; MOVQ xmm, m64; PACKSSWB/DW; PACKUSWB; PADDB/W/D; PADDSB/W; PADDUSB/W; PAND; PANDN; PCMPEQB/W/D; PCMPGTB/W/D; PMADDWD; PMULHW; PMULLW; POR; PSLLW/D/Q; PSRAW/D; PSRLW/D/Q; PSUBB/W/D; PSUBSB/W; PSUBUSB/W; PUNPCKHBW/WD/DQ; PUNPCKLBW/WD/DQ; PXOR.

V3. Direction Flag (DF) Mistakenly Denoted as a System Flag

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture* Section 3.4.3 "EFLAGS Register", in Figure 3-7 EFLAGS Register currently states:

X Direction Flag(DF)

It should state:

C Direction Flag(DF)

V4. Fopcode Compatibility Mode

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture* Section 8.1.8.1 "FOPCODE COMPATIBILITY MODE" currently states:

"When the FOP code compatibility mode is enabled, the IA32 architecture guarantees that if an unmasked x87 FPU floating-point exception is generated, the opcode of the last non-control instruction executed prior to the generation of the exception will be stored in the x87 FPU opcode register, and that value can be read by a subsequent FSAVE of FXSAVE instruction. When the fop compatibility mode is disabled (default), the value stored in the x87 FPU opcode register is undefined (reserved)."

It should state:

"If FOP code compatibility mode is enabled, the FOP is defined as it has always been in previous IA32 implementations (always defined as the FOP of the last non-transparent FP instruction executed before a FSAVE/FSTENV/FXSAVE).

If FOP code compatibility mode is disabled (default), FOP is only valid if the last non-transparent FP instruction executed before a FSAVE/FSTENV/FXSAVE had an unmasked exception."

V5. FCOS, FPTAN, FSIN, and FSINCOS Trigonometric Domain not Correct

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "INSTRUCTION REFERENCE" FCOS, FPTAN, FSIN, and FSINCOS trigonometric domain for C2 is incorrect. Under the FPU Flags affected, C2 currently states:

> C2 Set to 1 if source operand is outside the range -2^{63} to $+2^{63}$; otherwise, cleared to 0.

It should state:

> C2 Set to 1 if outside range -2^{63} <source operand $<+2^{63}$; otherwise, set to 0.

V6. Incorrect Description of Stack

The *Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture* Chapter 6, Section 6.2 paragraph 2, labeled "STACK" currently states:

> The next available memory location on the stack is called the top of stack. At any given time, the stack pointer (contained in the ESP register) gives the address (that is the offset from the base of the SS segment) of the top of the stack.

This paragraph is incorrect and will be removed from the section listed above.

V7. EFLAGS Register Correction

The *Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture* Section 3.7.2, Figure 3.7. "EFLAGS Register" currently states:

> Bit 11 "OF" as "X"

It should state:

> Bit 11 "OF" as "S"

V8. PSE-36 Paging Mechanism

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 3, Section 3.9, third paragraph currently states:

> As is shown in Table 3-3, the following flags must be set or cleared to enable the PSE-36 paging mechanism:
>
> - PSE-36 CPUID feature flag-When set, it indicates the availability of the PSE-36 paging mechanism on the IA-32 processor on which the CPUID instruction is executed.
> - PG flag (bit 31) in register CR0-Set to 1 to enable paging.
> - PSE flag (bit 4) in control register CR4 - Set to 1 to enable the page size extension for 4-Mbyte pages.
> - PAE flag (bit 5) in control register CR4-Clear to 0 to disable the PAE paging mechanism.

It should state:

> As is shown in Table 3-3, the following flags must be set or cleared to enable the PSE-36 paging mechanism:
>
> - PSE-36 CPUID feature flag-When set, it indicates the availability of the PSE-36 paging mechanism on the IA-32 processor on which the CPUID instruction is executed.
> - PG flag (bit 31) in register CR0-Set to 1 to enable paging.
> - PAE flag (bit 5) in control register CR4-Clear to 0 to disable the PAE paging mechanism.
> - PSE flag (bit 4) in control register CR4 and the PS flag in PDE- Set to 1 to enable the page size extension for 4-Mbyte pages.
> - Or the PSE flag (bit 4) in control register CR4- Set to 1 and the PS flag (bit 7) in PDE- Set to 0 to enable 4-KByte pages with 32-bit addressing (below 4GBytes).

V9. 0x33 Opcode

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Appendix A, Table A-2, the opcode corresponding to 0x33 currently states:

> Gb, Ev

It should state:

> Gv, Ev

Also, Page 3-791, XOR-Logical Exclusive OR, the two entries for opcode 33 currently states:

Opcode	Instruction	Description
33 /r	XOR r16,r/m16	r8 XOR r/m8

33 /r	XOR r32,r/m32	r8 XOR r/m8

It should state:

Opcode	Instruction	Description
33 /r	r16 XOR r/m16	r8 XOR r/m8
33 /r	r32 XOR r/m32	r8 XOR r/m8

V10. Incorrect Information for SLDT

In the *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference,* the Opcode/Instruction/Description table for SLDT currently states:

> "SLDT *r/m32* Store segment selector from LDTR in low-order 16 bits of *r/m32*"

It should state:

> "SLDT *r32* Store segment selector from LDTR in low-order 16 bits of *r32*."

V11. LGDT/LIDT Instruction Information Correction

In the *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference,* the sentence in the LGDT/LIDT instruction section currently states:

> "See 'SFENCE -- Store Fence' in this chapter for information on storing the contents of the GDTR and IDTR."

It should state:

> "See 'SGDT/SIDT' in this chapter for information on storing the contents of the GDTR and IDTR."

V12. Errors in Instruction Set Reference

The following changes will be made to The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference:*

1. *Page 3-586 "PMULUDQ—Multiply Packed Unsigned Doubleword Integers"* currently states:

   66 OF F4 /r PMULUDQ xmm1, xmm2/m128

   It should state:

   66 0F F4 /r PMULUDQ xmm1, xmm2/m128

2. *Page A-9, Table A-3. Two-byte Opcode Map:08H-7FH (First Byte is 0FH).*
Entry 2B currently states:

MOVNTPS
Wps, Vps
MOVNTPS (66)
Wpd, Vpd

It should state:

MOVNTPS
Wps, Vps
MOVNTPD (66)
Wpd, Vpd

3. *Page A-9, Table A-3, Two-byte Opcode Map:08H-7FH (First Byte is 0FH).*
Entry 3C currently states:

Blank (empty space)

It should state:

MOVNTI

4. *Page A-10, Table A-3, Two-byte Opcode Map:80H-7FH (First Byte is 0FH).*
Entry D7 currently states:

PMOVMSKB
Gd, Pq
PMOVMKSB (66)
Gd, Vdq

It should state:

PMOVMSKB
Gd, Pq
PMOVMSKB (66)
Gd, Vdq

5. *Page A-10, Table A-3, Two-byte Opcode Map:80H-7FH (First Byte is 0FH).*
Entry F7 currently states:

MASKMOVQ
Ppi, Qpi
MASKMOVQU (66)
Vdq, Wdq

It should state:

MASKMOVQ
Ppi, Qpi

MASKMOVDQU (66)
Vdq, Wdq

6. *Page A-11, Table A-3, Two-byte Opcode Map:88H-7FH (First Byte is 0FH).*
The title table currently states:

Table A-3. Two-byte Opcode Map:88H-7FH (First Byte is FFH)

It should state:

Table A-3. Two-byte Opcode Map:88H-7FH (First Byte is 0FH)

7. *Page A-11, Table A-3, Two-byte Opcode Map:88H-7FH (First Byte is 0FH).*
Entry FB currently states:

PSUBD
Pq, Qq
PSUBD (66)
Vdq, Wdq

It should state:

PSUBQ
Pq, Qq
PSUBQ (66)
Vdq, Wdq

8. *Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).*
Entry PMADD currently states:

PMADD – Packed Multiply add

It should state:

PMADDWD – Packed Multiply add

9. *Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).*
Entry PMULH currently states:

PMULH – Packed multiplication

It should state:

PMULHW – Packed multiplication, store high word

10. *Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).*
Add instruction PMULHUW :

PMULHUW – Packed multiplication, store high word (unsigned)

mmxreg2 to mmxreg1	0000 1111: 1110 0100: 11 mmxreg1 mmxreg2
memory to mmxreg	0000 1111: 1110 0100: mod mmxreg r/m

11. *Page B-21, Table B-12, MMX Instruction Formats and Encodings (Contd.).*
Entry PMULL currently states:

PMULL – Packed multiplication

It should state:

PMULLW – Packed multiplication, store low word

12. *Page B-40, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.*
Entry PMADD currently states:

PMADD – Packed multiply add

It should state:

PMADDWD – Packed multiply add

13. *Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.*
Entry PMULH currently states:

PMULH – Packed multiplication

It should state:

PMULHW – Packed multiplication, store high word

14. *Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.*
Add instruction PMULHUW:

PMULHUW – Packed multiplication, store high word (unsigned)

xmmreg2 to xmmreg1	0110 0110 : 0000 1111 : 11110 0100 : 11 xmmreg1 xmmreg2
memory to xmmreg	0110 0110 : 0000 1111 : 1110 0100 : mod xmmreg r/m

15. *Page B-41, Table B-19, Formats and Encodings of the SSE2 SIMD Integer Instruction.*
Entry PMULL currently states:

PMULL – Packed multiplication

It should state:

PMULLW – Packed multiplication, store low word

V13. RSM Instruction Set Summary

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture* Section 5.8 "INSTRUCTION SET SUMMARY" currently states:

RSM Return from system management mode (SSM)

It should state:

RSM Return from system management mode (SMM)

V14. Correct MOVAPS and MOVAPD Operand Section

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "INSTRUCTION REFERENCE" MOVAPS and MOVAPD operation section currently states:

Operation

DEST ← SRC;

It should state:

Operation

DEST ← SRC;

* #GP if SRC or DEST unaligned memory operand *;

V15. DAA—Decimal Adjust AL after Addition

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* page 3-173 currently states:

Operation

```
IF (((AL AND 0FH) > 9) or AF = 1)
  THEN
    AL ←AL + 6;
    CF ←CF OR CarryFromLastAddition; (* CF OR carry from AL ←AL + 6 *)
    AF ←1;
  ELSE
    AF ←0;
FI;
IF ((AL AND F0H) > 90H) or CF = 1)
  THEN
```

```
    AL ←AL + 60H;
    CF ←1;
  ELSE
    CF ←0;
FI;
```

It should state:

Operation

```
old_AL ← AL;
old_CF ← CF;
CF ← 0;
IF (((AL AND 0FH) > 9) or AF = 1)
  THEN
    AL ←AL + 6;
    CF ←old_CF or (Carry from AL ←AL + 6);
    AF ←1;
  ELSE
    AF ←0;
FI;
IF ((old_AL > 99H) or (old_CF = 1)
  THEN
    AL ←AL + 60H;
    CF ←1;
  ELSE
    CF ←0;
FI;
```

V16. DAS—Decimal Adjust AL after Subtraction

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* page 3-175 currently states:

Operation

```
IF (AL AND 0FH) > 9 OR AF = 1
  THEN
    AL ←AL - 6;
    CF ←CF OR CarryFromLastAddition; (* CF OR carry from AL ←AL - 6 *)
    AF ←1;
  ELSE AF ←0;
FI;
IF ((AL > 9FH) or CF = 1)
  THEN
    AL ←AL - 60H;
    CF ←1;
  ELSE CF ←0;
FI;
```

It should state:

Operation

```
old_AL ← AL;
old_CF ← CF;
CF ← 0;
IF (((AL AND 0FH) > 9) or AF = 1)
  THEN
    AL ← AL - 6;
    CF ← old_CF or (Borrow from AL ← AL - 6);
    AF ← 1;
  ELSE
    AF ← 0;
FI;
IF ((old_AL > 99H) OR (old_CF = 1))
  THEN
    AL ← AL - 60H;
    CF ← 1;
  ELSE
    CF ← 0;
FI;
```

V17. Omission of Dependency Between BTM and LBR

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter *15*, Section 5.3, page 15-15 currently states:

15.5.3. Monitoring Branches, Exceptions, and Interrupts (Pentium 4 and Intel Xeon Processors)

When the LBR flag in the IA32_DEBUGCTL MSR is set, the processor automatically begins recording branch records for taken branches, interrupts, and exceptions (except for debug exceptions)
in the LBR stack MSRs.

When the processor generates a debug exception (#DB), it automatically clears the LBR flag before executing the exception handler, but does not touch the LBR stack MSRs. The branch records for the last four taken branches, interrupts, and/or exceptions are thus retained for analysis by the debugger program.

The debugger can use the linear addresses in the LBR stack to reset breakpoints in the break-point-address registers (DR0 through DR3), allowing a backward trace from the manifestation
of a particular bug toward its source.

Before resuming program execution from a debug-exception handler, the handler must set the
LBR flag again to re-enable last branch recording.

It should state:

15.5.3. Monitoring Branches, Exceptions, and Interrupts (Pentium 4 and Intel Xeon Processors)

When the LBR flag in the IA32_DEBUGCTL MSR is set, the processor automatically begins

recording branch records for taken branches, interrupts, and exceptions (except for debug exceptions)
in the LBR stack MSRs.

When the processor generates a debug exception (#DB), it automatically clears the LBR flag before executing the exception handler. This action does not clear previously stored LBR stack MSRs. The branch record for the last four taken branches, interrupts and/or exceptions are retained for analysis.

A debugger can use the linear addresses in the LBR stack to reset breakpoints in the break-point address registers (DR0 through DR3). This allows a backward trace from the manifestation of a particular bug toward its source.

If the LBR flag is cleared and TR flag in the IA32_DEBUGCTLTR MSR remains set, the processor will continue to update LBR stack MSRs. This is because BTM information must be generated from entries in the LBR stack (see 14.5.5). A #DB does not automatically clear the TR flag.

V18. I/O Permissions Bitmap Base Addy > 0xDFFF Does not Cause #GP(0) Fault

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture,* page 12-6, section 12.5.2, last paragraph currently states:

> If the I/O bit map base address is greater than or equal to the TSS segment limit, there is no I/O permission map, and all I/O instructions generate exceptions when the CPL is greater than the current IOPL. The I/O bit map base address must be less than or equal to DFFFH.

It should state:

> If the I/O bit map base address is greater than or equal to the TSS segment limit, there is no I/O permission map, and all I/O instructions generate exceptions when the CPL is greater than the current IOPL.

V19. Wrong Field Width for MINSS and MAXSS

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 under "MAXSS—Return Maximum Scalar Single-Precision Floating-Point Value" page 3-415 currently states:

```
DEST[63-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]
```

It should state:

```
DEST[31-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]
```

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 under "MINSS—Return Minimum Scalar Single-Precision floating-Point Value" page 3-428 currently states:

DEST[63-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

It should state:

DEST[31-0] .IF ((DEST[31-0] = 0.0) AND (SRC[31-0] = 0.0)) THEN SRC[31-0]

V20. Figure 15-12 PEBS Record Format

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Section 15.9.6 " Programming the Performance Counters for Non-Retirement Events" page 15-37, Figure 15-12, first row currently states:

It should state:

V21. I/O Permission Bit Map

The *Intel Architecture Software Developer's Manual, Vol 1: Basic Architecture* Chapter 12, section 12.5.2 on Figure 12-2 (I/O Permission Bit Map) currently states:

Last byte of bit map must be followed by a byte with all bits.

It should state:

Last byte of bit map must be followed by a byte with all bits set.

Also, in the lower left hand corner of Figure 12-2 (I/O Permission Bit Map) currently states:

Last I/O base map must be

It should state:

Last I/O base map must be less than or equal to DFFFH

V22. Cache Description

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Table 3-10, the "sectored, 64 byte line size" description is used for the following descriptors: 0x22, 0x23, 0x79, 0x7a, 0x7b, 0x7c. This description will change to "dual-sectored line, 64 byte sector size" for clarity.

V23. Instruction Formats and Encoding

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Page B-8, CMOVcc memory to register should be encoded as "0000 1111 : 0100 tttn : mod reg r/m". Page B-8, CMP immediate with memory should be encoded as "1000 00sw : mod 111 r/m : immediate data". Page B-12 POP "segment register CS, DS, ES" should be encoded as "segment register DS, ES".

V24. Machine-Check Initialization

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Section 14.5 currently states*:*

14.5 MACHINE-CHECK INITIALIZATION

To use the processors machine-check architecture, software must initialize the processor to activate the machine-check exception and the error-reporting mechanism.

Example gives pseudocode for performing this initialization. This pseudocode checks for the existence of the machine-check architecture and exception on the processor, then enables the machine-check exception and the error-reporting register banks. The pseudocode assumes that the machine-check exception (#MC) handler has been installed on the system. This initialization procedure is compatible with the Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Following power up or power cycling, the IA32_MC*i*_STATUS registers are not guaranteed to have valid data until after the registers are initially cleared to all 0s by software, as shown in the initialization pseudocode in Example .

Machine-Check Initialization Pseudocode

```
EXECUTE the CPUID instruction;
READ bits 7 (MCE) and 14 (MCA) of the EDX register;
IF CPU supports MCE
    THEN
        IF CPU supports MCA
            THEN
                IF IA32_ MCG_CAP.MCG_CTL_P = 1
```

```
                (* IA32_MCG_CTL register is present *)
                    IA32_MCG_CTL ← FFFFFFFFFFFFFFFFH;
                    (* enables all MCA features *)
                FI;
                COUNT ← IA32_MCG_CAP.Count;
                MAX_BANK_NUMBER ← COUNT - 1;
                (* determine number of error-reporting banks supported *)
                IF (P6 Family Processor)
                    THEN
                        FOR error-reporting banks (1 through MAX_BANK_NUMBER) DO
                            IA32_MCi_CTL ← FFFFFFFFFFFFFFFFH;
                            (* enables logging of all errors except for MC0_CTL register *)
                        OD
                    ELSE (* Pentium 4 and Intel Xeon Processors *)
                        FOR error-reporting banks (0 through MAX_BANK_NUMBER) DO
                            IA32_MCi_CTL ← FFFFFFFFFFFFFFFFH;
                            (* enables logging of all errors including MC0_CTL register *)
                        OD
                FI;
                FOR error-reporting banks (0 through MAX_BANK_NUMBER) DO
                    IA32_MCi_STATUS ← 0000000000000000H; (* clears all errors *)
                OD
        FI;
    Set the MCE flag (bit 6) in CR4 register to enable machine-check exceptions;
FI;
```

It should state:

14.5 MACHINE-CHECK INITIALIZATION

To use the processors machine-check architecture, software must initialize the processor to activate the machine-check exception and the error-reporting mechanism.

Example gives pseudocode for performing this initialization. This pseudocode checks for the existence of the machine-check architecture and exception on the processor, then enables the machine-check exception and the error-reporting register banks. The pseudocode shown is compatible with the Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Following power up or power cycling, the IA32_MC*i*_STATUS registers are not guaranteed to have valid data until after the registers are initially cleared to all 0s by software, as shown in the initialization pseudocode in Example . In addition, when using P6 family processors, the software must set MCi_STATUS registers to 0 when doing a soft-reset.

Machine-Check Initialization Pseudocode

```
Check CPUID Feature Flags for MCE and MCA support
IF CPU supports MCE
```

```
THEN
    IF CPU supports MCA
    THEN
        IF (IA32_MCG_CAP.MCG_CTL_P = 1)
        (* IA32_MCG_CTL register is present *)
        THEN
            IA32_MCG_CTL ← FFFFFFFFFFFFFFFFH;
            (* enables all MCA features *)
        FI

        (* Determine number of error-reporting banks supported *)
        COUNT← IA32_MCG_CAP.Count;
        MAX_BANK_NUMBER ← COUNT - 1;

        IF (Processor Family is 6H)
        THEN
            (* Enable logging of all errors except for MC0_CTL register *)
            FOR error-reporting banks (1 through MAX_BANK_NUMBER)
            DO
                IA32_MCi_CTL ← 0FFFFFFFFFFFFFFFFH;
            OD

            (* Clear all errors *)
            FOR error-reporting banks (0 through MAX_BANK_NUMBER)
            DO
                IA32_MCi_STATUS ← 0;
            OD

        ELSE IF (Processor Family is 0FH) (*any Processor Extended Family *)
        THEN
            (* Enable logging of all errors including MC0_CTL register *)
            FOR error-reporting banks (0 through MAX_BANK_NUMBER)
            DO
                IA32_MCi_CTL ← 0FFFFFFFFFFFFFFFFH;
            OD

            (* BIOS clears all errors only on power-on reset *)
            IF (BIOS detects Power-on reset)
            THEN
                FOR error-reporting banks (0 through MAX_BANK_NUMBER)
                DO
                    IA32_MCi_STATUS ← 0;
                OD
            ELSE
                FOR error-reporting banks (0 through MAX_BANK_NUMBER)
```

```
                DO
                    (Optional for BIOS and OS) Log valid errors
                    (OS only) IA32_MCi_STATUS ← 0;
                OD

            FI
        FI
    FI

    Setup the Machine Check Exception (#MC) handler for vector 18 in IDT

    Set the MCE bit (bit 6) in CR4 register to enable Machine-Check Exceptions
FI
```

V25. Incorrect Description of MOVAPS #UD

The *Intel Architecture Software Developer's Manual, Vol 2: Instruction Set Reference* Section 3.2 "INSTRUCTION REFERENCE" MOVAPS #UD description is incorrect. Under Protected Mode Exceptions it currently states:

> #UD If CPUID feature flag SSE2 is 0.

It should state:

> #UD If CPUID feature flag SSE is 0.

Also, under Real-Address Mode Exceptions it currently states:

> #UD If CPUID feature flag SSE2 is 0.

It should state:

> #UD If CPUID feature flag SSE is 0.

V26. PSE-36 Paging Mechanism

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 5, Section 5.14, page 5-35 under “Interrupt 10 – Invalid TSS Exception (#TS)” the last paragraph of the description currently states:

> To insure that a valid TSS is available to process the exception, the invalid TSS exception must be a task called using a task gate.

It should state:

> The invalid-TSS handler must be a task called using a task gate. Handling this exception inside the faulting TSS context is not recommended and processor state may not be consistent.

V27. Segment Wraparound Compatibility

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 18, Section 18.29.1 (Segment Wraparound) has the following paragraph added at the end of the section:

The behavior when executing near the limit of a 4GB selector (limit=0xFFFFFFFF) is different between the Pentium Pro and the Pentium 4 family of processors. On the Pentium Pro, instructions which cross the limit -- for example, a two byte instruction such as INC EAX that is encoded as 0xFF 0xC0 starting exactly at the limit faults for a segment violation (a one byte instruction at 0xFFFFFFFF does not cause an exception). Using the Pentium 4 family, neither of these situations causes a fault.

V28. Performance Counting Clocks

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 15, Section 15.9.9 (Counting Clocks) will be updated to clarify the impact of Intel Hyper-Threading Technology when considering clock ticks and the Time Stamp Counter. A number of updates have been integrated into the section.

V29. New Cascading Counters Chapter

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Chapter 15, Section 15.9.6.6 will be updated to include information on extended cascading, a new feature included in the Intel NetBurst microarchitecture.

V30. Updated Table B-1 in Appendix B

The *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* Appendix B, a new column will be added to Table B-1“ MSRs in the Pentium 4 and Intel Xeon Processors”:

For Hyper-Threading Technology enabled processors, there are two logical processors per physical unit. If an MSR is Shared, this means that one MSR is shared between two logical processors. If an MSR is Unique, this means that each logical processor has its own MSR. The new column identifies the characteristics of each MSR.

V31. Message Signaled Interrupts

In the *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* section 8.11, information will be added to include Message Signaled Interrupts, an optional feature that enables PCI devices to request service by writing a system-specified message to a system-specified address (PCI DWORD memory write transaction).

V32. Selecting Memory Types for Pentium 4, Intel Xeon, and Pentium III Processors

In the *Intel Architecture Software Developer's Manual, Vol 3: System Programming Guide* chapter10, Section 10.5.2.2, Table 10.7, Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors currently states:

Table 10-7. Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors

MTRR Memory Type	PAT Entry Value	Effective Memory Type
UC	X	UC[1]
WC	UC	UC [2]
	UC-	WC
	WC	WC
	WT	Undefined
	WB	WC
	WP	Undefined
WT	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WT
	WP	Undefined
WB	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WB
	WP	WP
	UC	UC [2]
	UC	Undefined

	WC	WC
	WT	Undefined
	WB	WP
	WP	WP

NOTES:

1. The UC attribute comes from the MTRRs and the processors are not required to snoop their caches since the data could never have been cached. This attribute is preferred for performance reasons.
2. The UC attribute came from the page-table or page-directory entry and processors are required to check their caches because the data may be cached due to page aliasing, which is not recommended.

It should state:

Table 10-7. Effective Page-Level Memory Types for Pentium III, Pentium 4, and Intel Xeon Processors

MTRR Memory Type	PAT Entry Value	Effective Memory Type
UC	UC	UC[1]
	UC-	UC[1]
	WC	WC
	WT	UC[1]
	WB	UC[1]
	WP	UC[1]
WC	UC	UC[2]
	UC-	WC
	WC	WC
	WT	Undefined
	WB	WC
	WP	Undefined
WT	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WT
	WP	Undefined
WB	UC	UC [2]
	UC-	UC [2]
	WC	WC
	WT	WT
	WB	WB
	WP	WP

WP	UC	UC [2]
	UC-	Undefined
	WC	WC
	WT	Undefined
	WB	WP
	WP	WP

NOTES:

1. The UC attribute comes from the MTRRs and the processors are not required to snoop their caches since the data could never have been cached. This attribute is preferred for performance reasons.
2. The UC attribute came from the page-table or page-directory entry and processors are required to check their caches because the data may be cached due to page aliasing, which is not recommended.

SPECIFICATION CLARIFICATIONS

The Documentation Changes listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Numbers 251308)
- *Intel® Architecture Software Developer's Manual, Volumes 1, 2, and 3* (Order Numbers 243190, 243191, and 243192, respectively)

All Specification Clarifications will be incorporated into a future version of the appropriate Mobile Intel Celeron Processor on 0.13 Micron Process in Micro-FCPGA Package documentation.

There are no Specification Clarifications.

SPECIFICATION CHANGES

The Documentation Changes listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Numbers 251308)
- *Intel® Architecture Software Developer's Manual, Volumes 1, 2, and 3* (Order Numbers 243190, 243191, and 243192, respectively)

All Specification Changes will be incorporated into a future version of the appropriate Mobile Intel Celeron Processor on 0.13 Micron Process in Micro-FCPGA Package documentation.

There are no Specification Changes.